SF 298 MASTER COPY KEEP THIS COPY FOR REPRODUCTION PURPOSES

REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	July 1997	Technical - 97-15

4. TITLE AND SUBTITLE
Optimal line fitting using genetic algorithms

5. FUNDING NUMBERS
DAAH04-96-1-0082

6. AUTHOR(S)
Jennifer Pittman and C.A. Murthy

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES)
Center for Multivariate Analysis
Department of Statistics
417 Thomas Building
Penn State University
University Park, pA 16802

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
ARO 35518.21-MA

11. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12 b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Genetic algorithms are computational techniques which, given an optimization problem, use elements of directed and stochastic search to find the "best" solution from the space of potential solutions. We apply GA's to the problem of fitting the minimum least-squares piecewise linear function to a set of data points in R^2. We assume that the number of pieces is known but the knot locations are unknown. the effectiveness of our algprithm is demonstrated with two examples. Results are found to be quite promising and encourage further research.

14. SUBJECT TERMS
Genetic Algorithms, Splines, Neural Networks, Least Squares, Function Approximation

15. NUMBER IF PAGES
25

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

OPTIMAL LINE FITTING USING GENETIC ALGORITHMS

Jennifer Pittman and C.A. Murthy

Technical Report 97-15

July 1997

Center for Multivariate Analysis
417 Thomas Building
Penn State University
University Park, PA 16802

Research supported by the Army Research Office under Grant DAAHO4-96-1-0082. The United States Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

Optimal Line Fitting Using Genetic Algorithms

Jennifer Pittman and C. A. Murthy*

Center for Multivariate Analysis
Department of Statistics
326 Thomas Building
Pennsylvania State University
University Park, PA - 16802, USA

July 28, 1997

*On lien from the Machine Intelligence Unit, Indian Statistical Institute, 203 B.T. Road, Calcutta - 700035, India.

Summary

Functional approximation is a basic tool for characterizing and analyzing a process of interest. Data consisting of a set of input-output pairs $(x_i, y_i) \in \mathcal{R}^2,\ i = 1, \ldots, N$, are recorded and used to build a model of the corresponding process generating function. Such model construction is a common problem in various scientific fields, including pattern recognition, computer vision, and applied mathematics. The literature contains several methods for constructing such functions which involve building models from linear combinations of nonlinear functions. Examples of such methods include splines, kernel estimates, neural networks, and radial basis function networks (RBFNs).

Although these methods are commonly employed, they do have several significant drawbacks. For example, splines and kernel estimates require the estimation or approximation of critical parameters, either without guidelines or at computational expense. Neural networks and RBFNs have a tendency to overfit the data and their implementation often requires numerous adjustments supplied by an experienced user. It should also be noted that neural networks do not guarantee convergence to an optimal solution.

Genetic algorithms (GAs), on the other hand, have been proven to reach an optimal solution. GAs are recently developed search and optimization techniques which have been shown to be efficient, robust, and provide near optimal solutions. As such, GAs may represent a viable alternative to the above methods.

This paper proposes the use of GAs and least squares to fit piecewise linear functions to data sets in $\mathcal{R}^2$, where the optimal locations of the knots are unknown. A GA designed to perform such a task is described, along with supporting theory, and demonstrated on two datasets - one is fit with a single line (and the results compared to the least squares regression line) and the other is fit with a three-piecewise linear function. Our results show that, indeed, GAs can yield near optimal results at limited computational expense.

Several areas are available for future research. We are currently designing a genetic algorithm which determines the optimal number of lines as well as the knot placements. A comparison of the results of GAs to those of related methods, including those mentioned above, is also planned. Finally, we are exploring the use of GAs in multivariate situations, such as fitting hyperplanes to data, as an alternative to MARS and projection pursuit regression.

Abstract

Genetic algorithms are computational techniques which, given an optimization problem, use elements of directed and stochastic search to find the "best" solution from the space of potential solutions. We apply GA's to the problem of fitting the minimum least-squares piecewise linear function to a set of data points in $\mathcal{R}^2$. We assume that the number of pieces is known but the knot locations are unknown. The effectiveness of our algorithm is demonstrated with two examples. Results are found to be quite promising and encourage further research.

Key Words :

Genetic Algorithms, Splines, Neural Networks, Least Squares, Function Approximation

1 Introduction

Function approximation is a basic statistical tool for characterizing and analyzing some process of interest. Data or measurements, often subjected to random error, are recorded and an approximation of the process generating function is constructed from this partial information[1]. Constructing a function from a set of input-output pairs is a common problem in numerous scientific and engineering fields, including pattern recognition, computer vision, and applied mathematics[2,3]. The literature contains several methods for constructing such functions[3], including splines[1,4,5] and neural networks[2], using least-squares estimation. However, splines, kernel estimation, and related methods require the estimation or approximation of critical parameters either without guidelines or at computational expense[5,6]. Neural networks also suffer from this problem and do not guarantee convergence to an optimal solution[2,7]. As a consequence, function approximation is an area of ongoing research.

Genetic algorithms are recently developed search and optimization techniques from the field of artificial intelligence. They have been shown to be efficient, robust, and produce near-optimal solutions to problems in areas such as pattern recognition, machine learning, and statistical classification[8,9,10,11]. This paper proposes the use of genetic algorithms and least squares to fit piecewise linear functions to data sets in $\boldsymbol{R}^2$, where the optimal locations of the knots are unknown. We first present the problem and discuss current methods for function approximation. We then introduce genetic algorithms and detail how GA's can be used to fit optimal piecewise-linear functions. Several examples are presented with results, and areas for future research are mentioned.

2 Problem Statement and Current Methodology

We are given data $(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$, $(x_i, y_i) \in \mathcal{R}^2 \; \forall \; i = 1, \ldots, N$, $1 \leq N < \infty$. Define $x_{(1)} = \min\{x_i;\ i = 1, \ldots, N\}$, $x_{(N)} = \max\{x_i;\ i = 1, \ldots, N\}$. The values x_i and y_i are related by an unknown function f such that $y_i = f(x_i) + \epsilon_i$, where ϵ_i is a random error. The problem is to approximate the function f by a k-piecewise linear function $\hat{f}$, k known, where the knot locations $\boldsymbol{z} = (z_1, \ldots, z_k)$ are unknown and the least-squares error is to be minimized. We make no assumptions regarding the smoothness of $\hat{f}$ or the distributions of $(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{\epsilon}$.

Classical approximation theory suggests several methods for solving such a problem, methods which involve building models from linear combinations of nonlinear functions[12]. Such linear estimators can be expressed as

$$\hat{f}(x_i) = \sum_{i=1}^{n} K_\lambda(x, x_i) y_i \tag{1}$$

where $K_\lambda(x, x_i)$ is a weighting function which depends on some parameter(s) λ[6]. Some examples includes kernel estimates, series approximation (which we will not explicitly discuss), and spline fitting[1,4,5,6], as well as the more recent neural network and radial basis function estimates[12,13,14].

Kernel estimators can be expressed in the above form where the weighting function or kernel K_λ has a simple form independent of the design of $\boldsymbol{x}$. Examples of such weighting functions are the uniform and triangular kernels. K_λ is a bounded function assumed to have support [-1,1], with a maximum at zero, and is usually chosen to satisfy certain moment conditions[6]. The choice of moment conditions determines the order m of the kernel estimate: conditions on higher order moments lead to higher order estimates. The parameter λ is called the *bandwidth* and determines (1) the maximum distance away from x_i a data point can be and still be included in the estimation of $f(x_i$, and (2) the amount of emphasis placed on observations at certain distances from x_i. Note that kernel methods require the researcher to select the appropriate values for K_λ, λ, and m. Although these choices are critical to the quality of the resulting estimate, they are often made by trial-and-error or time consuming adaptive methods[15]. λ is often chosen using *cross-validation*, although CV does not guarantee the selection of the optimal λ[6].

Another class of linear estimators closely related to kernel methods are *splines*. A spline of order L with knots at $z_1, \ldots, z_K$ is a function s of the form

$$s(x) = \sum_{i=0}^{L-1} \theta_i x^i + \sum_{i=1}^{K} \delta_i (x - z_i)^{(L-1)} \tag{2}$$

for $\theta_i \in \mathcal{R}$, $i = 0, \ldots, L-1$, and $\delta_i \in \mathcal{R}$, $i = 1, \ldots, K$. In other words, a spline is a piecewise polynomial where the pieces are tied at knots in such a way that s satisfies certain continuity properties (e.g., the first $L-1$ derivatives are continuous). As such they can be viewed as an extension of polynomial regression[6]. Different classes of splines can be formed by using different basis functions, e.g., B-splines, periodic splines, etc.[1]. Splines are useful when we want an estimate which meets a fitness criterion as well as a smoothness criterion. Hence we may estimate y by choosing $\hat{f}$ to minimize

$$n^{-1} \sum_{j=1}^{N} (y_j - f(x_j))^2 + \lambda \int_a^b f^{(m)} dx \tag{3}$$

for $\lambda > 0$, $m \in \mathcal{Z}^+$, and $a \leq x_j \leq b \ \forall \ j = 1, \ldots, N$. The solution $\hat{f}$ is called a *smoothing spline* estimate, and λ is the *smoothing parameter*. λ determines the tradeoff between goodness-of-fit and smoothness[15]. Splines have applications in areas such as computer tomography[6] and military analysis[4].

To use (smoothing) splines for analysis, the order L of the spline, the number and location of the knots, and λ need to be determined (as well as the choice of basis and the smoothing criterion). Finding a good estimate for λ is computationally demanding[15] and m is often based on prior information[6] as opposed to theoretical considerations. Schwetlick and Schütze[5] describe an algorithm which optimizes the location and number of 'free' knots but is computationally 'too expensive' and involves the approximation of various parameters whose effects on the final estimate are unknown. Larson[4] finds a closed form for the minimizing abscissa for unknown knot locations, but does not mention the optimization of the number of knots.

A recent development in functional approximation is the use of *neural networks* (NN) and *radial basis functions* (RBFs). Multilayer neural networks are linear (in the sense

of (1)) function approximators of the form, e.g.,

$$\hat{f}(x_k, \mathcal{W}) = \sum_{j=1}^{M} \beta_j g_j(\alpha_j x_k) \tag{4}$$

where $\beta_j, 1 \leq j \leq M$ are the weights connecting M hidden units to the output unit, $\alpha_j, 1 \leq j \leq M$ are weights connecting the input layer unit to the jth hidden layer unit, and the g_j's are the hidden layer activation functions[12]. $\mathcal{W}$ is the matrix of network weights. A special case is based on radial basis functions where the approximation is produced by passing each x_i through a set of basis functions, each containing a RBF center, multiplying the result by a coefficient, and then summing the results. In other words,

$$\hat{f}(x_k, \mathcal{W}) = w_0 + \sum_{j=1}^{M} w_j \tau^{-p} \phi(\|x_k - c_j\|/\tau) \tag{5}$$

where ϕ is the radial basis function, $\{c_j\}$ is the set of RBF centers, and τ is a scale parameter. Often ϕ corresponds to a Gaussian density[2,13]. Note that a radial basis function network (RBFN) is essentially a kernel method for regression[2]. NNs are easily programmed and, as a result, have become an almost universal optimization 'crank': simply toss in the data, add any number of parameters, and wait for gradient descent to produce the result. NNs do require numerous adjustments, supplied by an experienced user, and do have a tendency to overfit or overparameterize the data[7,13]. They may also get stuck in local minima, unlike GA's[2,16]. Chen and Jain[12] report that backward propagation can be slow and sensitive to noise. They suggest a robust modification whose parameters are the focus of further study. RBFNs have been shown to outperform MLPs[13] even though the choice of centers[2] and the curse of dimensionality can make implementation difficult. It should also be noted that both NN and RBFN results lack interpretability[7].

Our preliminary studies indicate that GA's may represent a viable alternative to the above methods.

3 Genetic Algorithms

Genetic algorithms are stochastic search methods which provide a near optimal solution to the evaluation function of an optimization problem[8,9,10,11,16]. They can be used to search complex, multimodal surfaces via steps based on the processes of natural genetic systems. They are designed to work simultaneously on a group of possible solutions (parallelism) which helps prevent the algorithm from getting stuck in a local optimum. Their effectiveness has been shown in numerous problem solving applications, including scheduling, classifier systems, and pattern recognition[11].

Each possible solution is encoded as a string or chromosome; a set of such chromosomes is called a *population*. An evaluation (*fitness*) function provides a mapping from the chromosome space to the solution space. GA's start with an initial population of a fixed number of randomly generated strings. At each iteration, three basic operations - selection, crossover, and mutation - are applied over the current population to yield

a new population of strings. This cycle is repeated until some termination criterion is achieved, at which time the best string achieved is generally taken as the solution to the optimization problem.

3.1 Example

For a more detailed look at this process, we will detail the stages of a GA model, the **elitist** model. Consider the problem of maximizing a function $f(x)$, $x \in D$, where D is a finite set and $f(x) > 0 \; \forall \; x \in D$. Each string S, built from members of a finite alphabet $\mathcal{A} = \{\alpha_1, \ldots, \alpha_a\}$, corresponds to a value x in D and may be written as

$$S = (\gamma_0, \gamma_1, \ldots, \gamma_L); \;\; \gamma_i \in \mathcal{A} \;\; \forall \, i = 0, \ldots, L$$

The number of different strings that are possible is a^L. A random sample of size M (even) is drawn from these a^L possible strings with replacement to form the initial population, $\mathcal{Q}$. The evaluation or fitness value of each string S is $fit(S) = f(x)$ where $x \in D$ is the value represented by S.

The first operation, *selection*, is modeled after Darwin's concept of 'survival of the fittest'. Strings from the population are selected and placed in a mating pool; the probability of selection for string j is $b_j = fit(S_j) / \sum_{i=1}^{M} fit(S_i)$. For example, if $B_j = \sum_{i=1}^{j} b_k$, M strings are selected and placed in the mating pool by the following process:

1. Generate a random number rnd_i from [0,1]
2. If $rnd_i \leq B_1$, select S_1; for $j = 2, \ldots, M$, if $B_{j-1} < rnd_i \leq B_j$, select S_j

Note that strings with low fitness values are rarely selected while some strings may be selected more than once. We denote the mating pool, our new population, as $\mathcal{Q}_1$.

In *single point crossover*, or *reproduction*, pairs of strings exchange information, thereby generating two new offspring for the next population. All strings are paired at random in such a way that each string belongs to only one pair (hence there are $M/2$ pairs). Let the given pair be denoted as

$$\beta = (\beta_1, \ldots, \beta_L) \;\text{ and }\; \tau = (\tau_1, \ldots, \tau_L)$$

and let p_c be the probability that a given pair of strings undergoes crossover. Then the crossover operation on a given pair may be described as

1. Generate a random number rnd from [0,1]
2. If $rnd \leq p_c$, then generate a random integer pos from [1, L-1].
3. Strings β and τ are replaced by strings β' and τ' where

$$\beta' = (\beta_1, \ldots, \beta_{pos}, \tau_{pos+1}, \ldots, \tau_L) \;\text{ and }\; \tau' = (\tau_1, \ldots, \tau_{pos}, \beta_{pos+1}, \ldots, \beta_L)$$

The resulting population is denoted $\mathcal{Q}_2$. Note that $\mathcal{Q}_2$ has M strings, some of which may have also been elements of $\mathcal{Q}_1$.

Mutation involves the random altering of characters in the chromosomes (strings) of $\mathcal{Q}_1$. Let p_m denote the probability of mutation of a given character. Then, for each character β_i of every string β, the mutation stage consists of

1. Generate a random number rnd from [0,1]
2. If $rnd \leq p_m$, mutate character β_i by replacing it at random with an element from $\mathcal{A} - \{\beta_i\}$.

Note that through mutation, a given string can become any of the a^L possible strings. The mutation probability may vary over iterations, initially taking a high value, then decreasing to a pre-specified minimum, then increasing again in the later stages of the algorithm. When the algorithm has little knowledge of the search space, the algorithm is encouraged to explore it's domain through a high mutation probability. As the number of iterations increases the algorithm will move towards a solution, hence the mutation probability is decreased to allow a search of the vicinity near this solution. To avoid the convergence of the algorithm to a local optima, the mutation probability is increased in the later stages to again allow for a more random search. The resulting population we denote as $\mathcal{Q}_3$.

We now replace our initial $\mathcal{Q}$ with $\mathcal{Q}_3$ and repeat the above stages until the algorithm converges to a satisfactory solution. The stages we have discussed so far are common to all GA models. In the elitist model of GA's (EGA), a further operation, *elitism*, is added to ensure that knowledge about the best string obtained so far is preserved. In this way the algorithm can report at any time the best solution achieved during the entire process. The basic steps of the elitist model are

1. Generate an initial population Q and find the fitness values of each string S in Q.
2. Find the string S_{maxQ} in Q with the maximum fitness value fit_{maxQ} of all of the strings in Q
3. Perform selection on Q yielding Q_1
4. Perform crossover on Q_1 yielding Q_2
5. Perform mutation on Q_2 yielding Q_3
6. (*elitism*) Compare the fitness value of each string in Q_3 with the fitness value of S_{maxQ}. If no string in Q_3 has a fitness value greater than or equal to fit_{maxQ}, replace the worst string in Q_3 with S_{maxQ}.
7. Replace Q with Q_3 and go to step 2.

3.2 Remarks

3.2.1 Stopping Rules and Convergence

With any optimization technique, it is important to ensure that the process will lead to the optimal solution. It has been theoretically proven[16] that elitist genetic algorithms will converge to the optimal solution as the number of iterations, n, goes to infinity. However, in practice, n is finite so a stopping rule is used to determine when the algorithm has reached an acceptable solution. There is, in general, no stopping rule in the literature which will ensure the convergence of GA's to the optimal solution. Two common stopping rules are

- Execute the process for a fixed number of iterations and report the best string found as the solution.
- Execute the process until the fitness value does not show adequate improvement over a fixed number of iterations, and report the best string found as the solution.

The rate of convergence of GA's depends on M, p_c, and p_m. Hence the values of these parameters must be chosen properly. Note, however, that the proof of convergence to the optimal solution does not depend on the parameter values, i.e., the GA will converge to the optimum as n goes to infinity regardless of the parameter values chosen.

3.2.2 Pattern Classification

Recently, several applications of genetic algorithms in the field of pattern classification have been reported[8,10,11]. Classification is the problem of finding a decision boundary that can correctly distinguish between different classes in the feature space. Given a set of data points in $\mathcal{R}^N$, $N \geq 1$, genetic algorithms can be used to perform this task by, for example, allowing each string to represent a decision boundary formed by a set of lines or hyperplanes. A fitness function which takes larger values for smaller numbers of misclassifications is then maximized. Usually the optimal decision boundary is nonlinear so our task is to approximate the optimal boundary with a set of linear segments. The algorithm is run until a decision boundary with an acceptable number of misclassifications is found.

The application of GA's for classification is similar to the application discussed in this paper. Here, each string also represents a set of lines and the string which best approximates the optimal solution is reported as the result. Our interest, however, is focused on finding a piecewise linear function which will minimize the squared distance of the data points from the function, and **not** on dividing the data into distinct classes.

4 Theory of Line Fitting in $\mathcal{R}^2$

4.1 Mathematical Formulation

Let $(\boldsymbol{x}, \boldsymbol{y})$ be the given data set, $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)$, $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)$, $(x_i, y_i) \in \mathcal{R}^2 \ \forall \ i = 1, \ldots, N, \ 1 \leq N < \infty$. Define $x_{(1)} = \min\{x_i, \ i = 1, \ldots, N\}$, $x_{(N)} = \max\{x_i, \ i = 1, \ldots, N\}$, $y_{(1)} = \min\{y_i, \ i = 1, \ldots, N\}$, $y_{(N)} = \max\{y_i, \ i = 1, \ldots, N\}$. Let $\mathcal{L}_{k_0}$ represent the class of all k_0-piecewise linear functions $L_{jk_0}(\boldsymbol{x})$ in $\mathcal{R}^2$ that can be expressed in the following form:

$$L_{jk_0}(\boldsymbol{x}) = \begin{cases} L_{j_1 k_0}(x_i) & \text{if } x_{(j1)} \leq x_i < x_{(j2)} \\ L_{j_2 k_0}(x_i) & \text{if } x_{(j2)} \leq x_i < x_{(j3)} \\ \vdots \qquad\qquad & \vdots \qquad\quad \vdots \\ L_{j_{k_0} k_0}(x_i) & \text{if } x_{(jk_0)} \leq x_i \leq x_{(j(k_0+1))} \end{cases} \tag{6}$$

where $x_{(j1)} \leq x_{(j2)} \leq \cdots \leq x_{(jk_0)} \leq x_{(j(k_0+1))}$, $x_{(j1)} = x_{(1)}$, $x_{(j(k_0+1))} = x_{(N)}$, and each $L_{j_i k_0}$, $i = 1, \ldots, k_0$, can be expressed as

$$x \ \cos\theta_{ji} + y \ \sin\theta_{ji} = d_{ji}, \ \ 0 \leq \theta_{ji} \leq \pi, \ \ d_{ji} \in \mathcal{R}$$

where θ_{ji} $(0 \leq \theta_{ji} < \pi)$ is the polar angle formed when the polar axis is the y-axis and the origin is the intersection point between the y-axis and $L_{j_i k_0}$, and d_{ji} is the perpendicular distance of the line from the origin (0,0). The number of elements $L_{jk_0} \in \mathcal{L}_{k_0}$ is uncountable. However, we can restrict the class of functions under consideration to a finite (discrete) set by restricting the values of θ and d. Let $\mathrm{l_a}$ be the number of bits used to express θ and let $\mathrm{l_d}$ be the number of bits used to express d. Note that the precision of the line is determined by both $\mathrm{l_a}$ and $\mathrm{l_d}$. We restrict θ_{ji} to the values $\{0, \frac{\pi}{2^{\mathrm{l_a}}}, \frac{2\pi}{2^{\mathrm{l_a}}}, \ldots, \frac{(2^{\mathrm{l_a}}-1)\pi}{2^{\mathrm{l_a}}}\}$. In specifying d_{ji}, we utilize the rectangle *rect* formed by the points $(x_{(1)}, y_{(1)})$, $(x_{(N)}, y_{(1)})$, $(x_{(1)}, y_{(N)})$ and $(x_{(N)}, y_{(N)})$. Note that *rect* contains the entire data set. Let *diag* be the maximum diagonal of *rect* and let l_θ be defined as

$$l_\theta = \begin{cases} x_{(1)} \cos\theta + y_{(1)} \cos\theta & \text{if } 0 \leq \theta < \pi/2 \\ x_{(N)} \cos\theta + y_{(1)} \cos\theta & \text{if } \pi/2 \leq \theta < \pi \end{cases} \tag{7}$$

Then for a given θ_{ji}, d_{ji} may only take values within the set $\{d_{ji} = l_{\theta_{ji}} + k_{ji}\delta : \ k_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l_d}} - 1\}, \ \delta = diag/(2^{\mathrm{l_d}} - 1)\}$. Let $\mathcal{L}^0_{k_0}$ denote the finite set of functions in $\mathcal{L}_{k_0}$ which satisfy these restrictions. Then $\mathcal{L}^0_{k_0}$ may be expressed as

$$\begin{aligned} \mathcal{L}^0_{k_0} = \{ & L_{jk_0}(\boldsymbol{x}) : L_{jk_0}(\boldsymbol{x}) \ \in \ \mathcal{L}_{k_0}, \\ & L_{j_i k_0} \text{ is of the form } x\cos\theta_{ji} + y\sin\theta_{ji} = l_{\theta_{ji}} + k_{ji}\delta \ \ \forall \ \ i = 1, \ldots, k_0, \\ & \theta_{ji} \in \{0, \tfrac{\pi}{2^{\mathrm{l_a}}}, \tfrac{2\pi}{2^{\mathrm{l_a}}}, \ldots, \tfrac{(2^{\mathrm{l_a}}-1)\pi}{2^{\mathrm{l_a}}}\}, \ k_{ji} \in \{0, 1, \ldots, 2^{\mathrm{l_d}} - 1\}, \text{ and } \delta = diag/(2^{\mathrm{l_d}} - 1)\}. \end{aligned}$$

Note 1 A line with $d = l_\theta$ intersects *rect* at the point $(x_{(1)}, y_{(1)})$, if $0 \leq \theta \leq \pi/2$, or the point $(x_{(N)}, y_{(1)})$, if $\pi/2 \leq \theta < \pi$. The parameter $k_{ji}\delta$, $0 \leq k_{ji}\delta \leq diag$, is sometimes referred to as the $offset$ value.

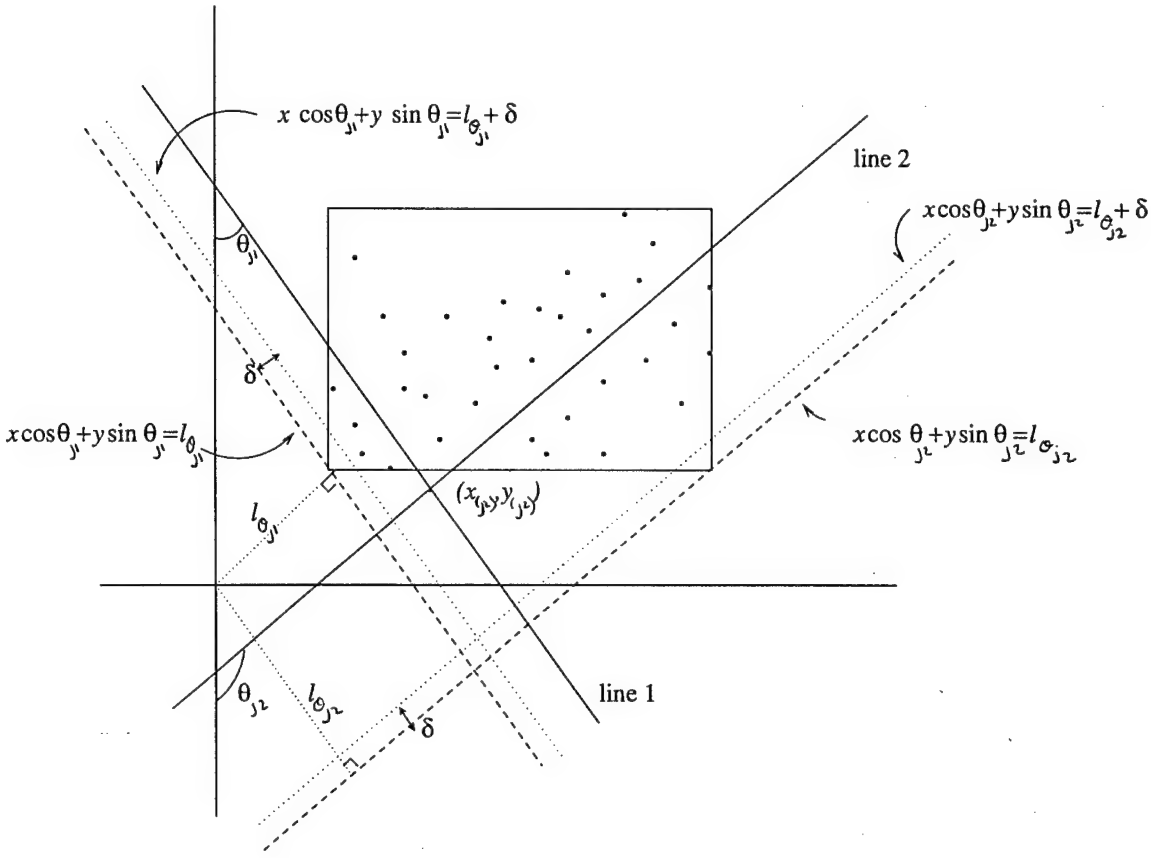


Figure 1: A data set with functions from $\mathcal{L}_2^0$

Note 2 For fixed i and θ_{ji} the lines $L_{j_i k_0}$, $k_{ji} = 0, \ldots, 2^{l_d} - 1$, are parallel and evenly spaced across the area covered by *rect*.

Note 3 If $l_a^* \geq l_a$ and $l_d^* \geq l_d$, then $\mathcal{L}_{k_0}^0 \subset \mathcal{L}_{k_0}^{0*}$ where $\mathcal{L}_{k_0}^0$ corresponds to l_a and l_d and $\mathcal{L}_{k_0}^{0*}$ corresponds to l_a^* and l_d^*.

Figure 1 represents a sample data set with several functions from $\mathcal{L}_{k_0}^0$. For sake of clarity, we will henceforth specify $\mathcal{L}_{k_0}^0$ as $\mathcal{L}_{k_0}^0(\Theta, \mathcal{K})$ where θ has Θ possible values and k has $\mathcal{K}$ possible values (*note that* $\Theta = 2^{l_a}$ *and* $\mathcal{K} = 2^{l_d}$) and specify $L_{jk_0}(\boldsymbol{x})$ as $L(\boldsymbol{\theta}_{jk_0}, \boldsymbol{k}_{jk_0})$. Our goal is to use genetic algorithms to find the minimum least-squares k_0-piecewise linear function where k_0 is known. This is possible if and only if

1. Our search space $\mathcal{L}_{k_0}^0(\Theta, \mathcal{K})$ contains the optimal solution, i.e., minimum least-squares function.
2. The algorithm converges to this optimal solution

as $\Theta \to \infty$ and $\mathcal{K} \to \infty$. We first determine whether these conditions are met when $k_0 = 1$.

4.2 Case $k_0 = 1$

In the case where $k_0 = 1$, we would like our optimal string to represent the minimum least-squares line, i.e., the line whose fitted values $\hat{\mathbf{y}}_0$ satisfy

$$(\sum_{i=1}^{N}(\hat{y_{0i}} - y_i)^2)^{-1} = \max_j\{(\sum_{i=1}^{N}(\hat{y_{ji}} - y_i)^2)^{-1} : \hat{\mathbf{y}}_j = L(\theta_{j1}, k_{j1})|_{\mathbf{x}},\ L_{j1} \in \mathcal{L}_1\} \qquad (8)$$

The least-squares line is known as $\hat{\mathbf{y}} = \hat{\beta}_1\mathbf{x} + \hat{\beta}_0$, where

$$\hat{\beta}_1 = \frac{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x} \qquad \bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i \qquad \bar{y} = \frac{1}{N}\sum_{i=1}^{N}y_i$$

[?] Note that the least-squares line intersects *rect* since it passes through the point $(\bar{x}, \bar{y})$.

Let

$$\mathcal{L}_1^0(\Theta, \mathcal{K}) = \{L(\theta_{j1}, k_{j1}) : L(\theta_{j1}.k_{j1}) \in \mathcal{L}_1,\ L(\theta_{j1}, k_{j1}) \text{ is of the form}$$
$$x\cos\theta_{j1} + y\sin\theta_{j1} = l_{\theta_{j1}} + k_{j1}\delta, \text{ where}$$
$$\theta_{j1} \text{ is one of } \Theta \text{ values, } k_{j1} \text{ is one of } \mathcal{K} \text{ values}\}$$

and let

$$\mathcal{B}_1 = \{L(\theta_{m1}, k_{m1}) : L(\theta_{m1}, k_{m1}) \in \mathcal{L}_1,\ \exists\ (x_r, y_r) \in \textit{rect} \text{ satisfying } L(\theta_{m1}, k_{m1}),$$
$$0 \le \theta_{m1} < \pi,\ k_{m1} \in \mathcal{R}\}.$$

Figures 2 and 3 show lines from $\mathcal{L}_1^0(\Theta, \mathcal{K})$ for a sample data set. We shall prove that our class $\mathcal{L}_1^0(\Theta, \mathcal{K})$ will contain the least-squares line as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.

For simplicity, let $\rho = \theta + k\delta$ for given θ and k.

Proposition 4.1 Let $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$. Let $\epsilon > 0$. Then $\exists\ (\Theta_\epsilon, \mathcal{K}_\epsilon)$: $\forall\ \Theta > \Theta_\epsilon$ and $\mathcal{K} > \mathcal{K}_\epsilon$, $\exists\ L(\theta, k)$*:*

1. $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})$
2. $\mid \theta - \theta_{m1} \mid < \epsilon/2$ and $\mid \rho_{m1} - \rho \mid < \epsilon/2$

Proof: Let $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$ and $\epsilon > 0$ be given. Choose $\Theta_\epsilon : \pi/2^{l_a} = \pi/\Theta_\epsilon < \epsilon/2$. Similarly, choose $\mathcal{K}_\epsilon$: $\delta = diag/(2^{l_d} - 1) = diag/(\mathcal{K}_\epsilon - 1) < \epsilon/2$. Then $\exists\ L(\theta, k) \in \mathcal{L}_1^0(\Theta_\epsilon, \mathcal{K}_\epsilon)$: *2.* is satisfied. By Note 3 above, if $L(\theta, k) \in \mathcal{L}_1^0(\Theta_\epsilon, \mathcal{K}_\epsilon)$, then $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})\ \forall\ \Theta > \Theta_\epsilon$ and $\forall\ \mathcal{K} > \mathcal{K}_\epsilon$. Hence *1.* is satisfied. ♠

Proposition 4.2 For each $\epsilon > 0$, $\exists\ (\Theta_\epsilon, \mathcal{K}_\epsilon)$: for all $\Theta > \Theta_\epsilon$ and for all $\mathcal{K} > \mathcal{K}_\epsilon$, given any $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$, $\exists\ L(\theta, k)$:

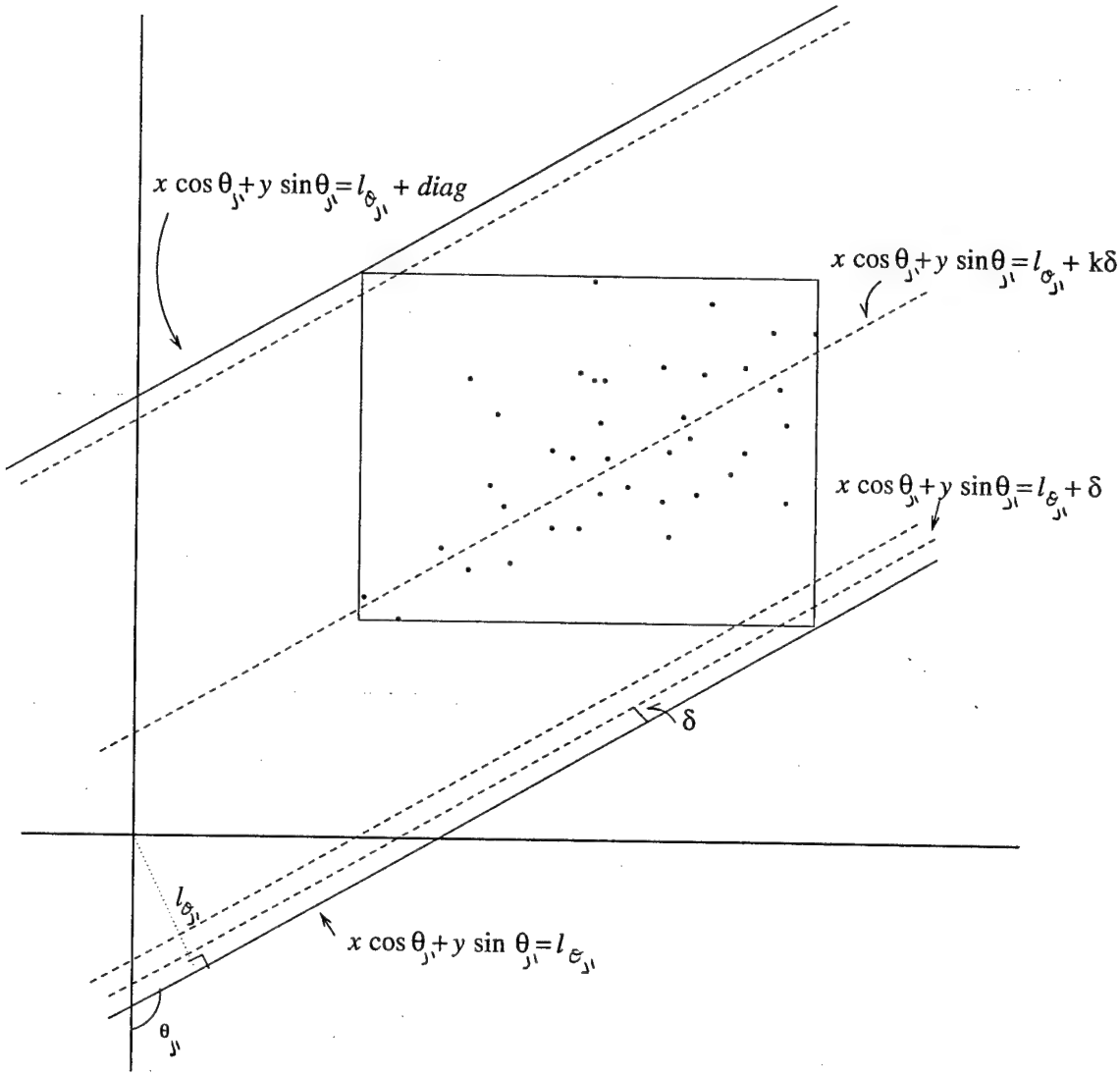


Figure 2: A data set with lines from $\mathcal{L}_1^0(\Theta, \mathcal{K})$, $\theta_{j1} > \pi/2$

1. $L(\theta, k) \in \mathcal{L}_1^0(\Theta, \mathcal{K})$
2. $\mid \theta - \theta_{m1} \mid < \epsilon/2$ and $\mid \rho_{m1} - \rho \mid < \epsilon/2$

Proof: Let $\epsilon > 0$ be given. Choose $\Theta_\epsilon : \pi/2^{l_a} = \pi/\Theta_\epsilon < \epsilon/2$. Then for $\theta_{\epsilon_{(i)}} \in \{0, \ldots, \frac{(2^{\Theta_\epsilon}-1)\pi}{2^{\Theta_\epsilon}};\ \theta_{\epsilon_{(i-1)}} \leq \theta_{\epsilon_{(i)}}\ \forall\ i,\ i = 1, \ldots, \Theta_\epsilon\}$, we have $\mid 0 - \theta_{\epsilon_{(1)}} \mid < \epsilon/2$, $\mid \theta_{\epsilon_{(1)}} - \theta_{\epsilon_{(2)}} \mid < \pi/2, \ldots, \mid \theta_{\epsilon_{(\Theta_\epsilon)}} - \pi \mid < \pi/2$. So given any $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$ we can choose Θ_ϵ so that $\exists\ \theta_{\epsilon_{m1}} \in \{0, \ldots, \frac{(2^{\Theta_\epsilon}-1)\pi}{2^{\Theta_\epsilon}}\} : \mid \theta_{m1} - \theta_{\epsilon_{m1}} \mid < \epsilon/2$.

For any angle $\theta_{\epsilon_n} \in [\frac{n\pi}{\Theta_\epsilon}, \frac{(n+1)\pi}{\Theta_\epsilon}]$, $n = 0, \ldots, \Theta_\epsilon - 2$, the corresponding $\rho_{\epsilon_n} \in [\gamma_{\epsilon_{n_1}}, \gamma_{\epsilon_{n_2}}]$, $l_{\theta_{\epsilon n}} \leq \gamma_{\epsilon_{n_1}} \leq \gamma_{\epsilon_{n_2}} \leq diag$. Find

$$\nu = \max_n \{\sup_{\epsilon_n} \{[\gamma_{\epsilon_{n_2}} - \gamma_{\epsilon_{n_1}}],\ n = 0, \ldots, \Theta_\epsilon - 2\}\} \tag{9}$$

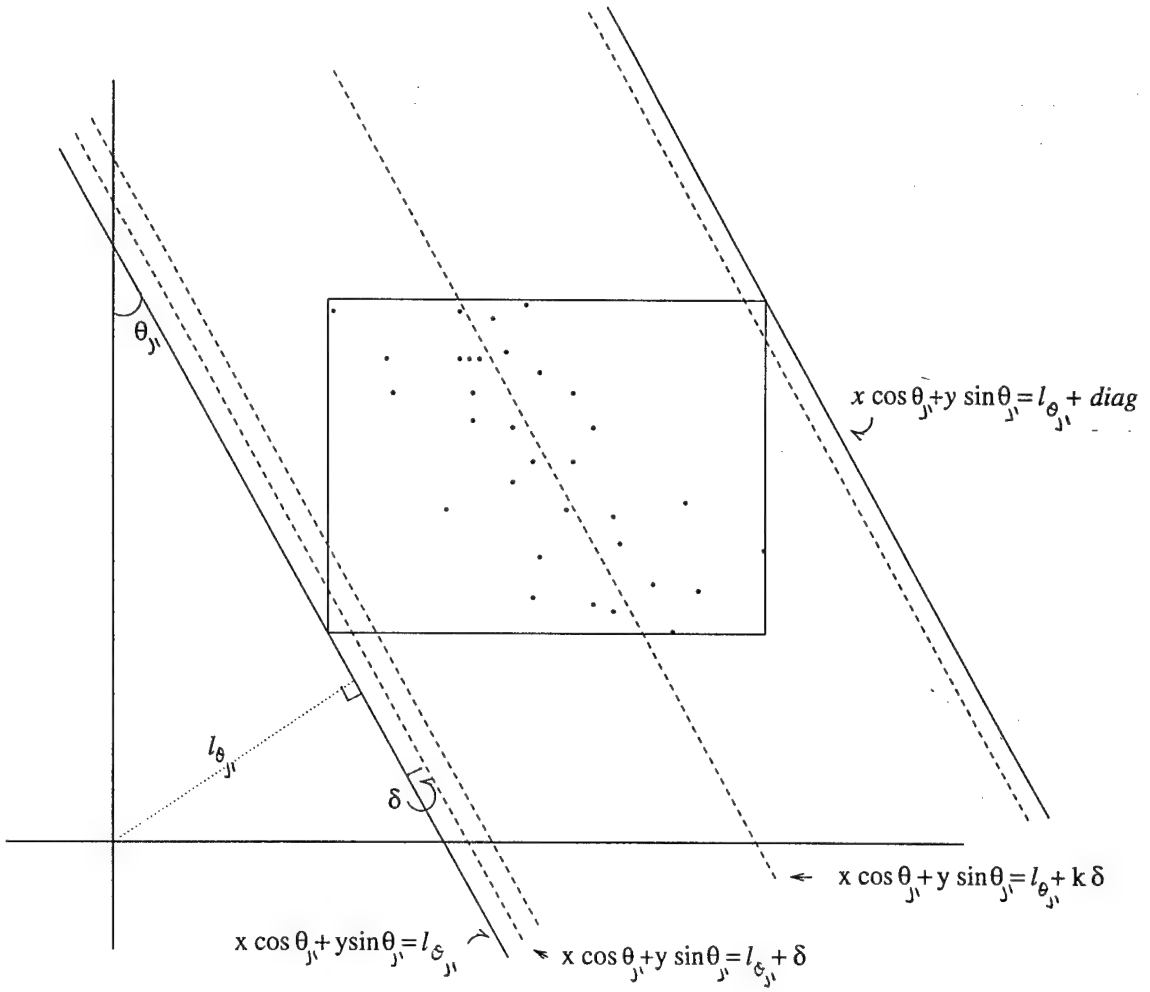


Figure 3: A data set with lines from $\mathcal{L}_1^0(\Theta, \mathcal{K})$, $\theta_{j1} < \pi/2$

Choose $\mathcal{K}_\epsilon$: $\nu/\mathcal{K}_\epsilon < \epsilon/4$ and $\mathcal{K}_\epsilon = 2^y$ for some $y \in \mathcal{R}$. Then given any $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$ we can choose $\mathcal{K}_\epsilon$ so that $\exists\ k_{\epsilon_{m1}} \in \{0, \ldots, 2^{\mathcal{K}_\epsilon} - 1\} : \mid \rho_{\epsilon_{m1}} - \rho_{m1} \mid < \epsilon/2$.

Hence given any $\epsilon > 0$ and $L(\theta_{m1}, k_{m1}) \in \mathcal{B}_1$ we can find Θ_ϵ and $\mathcal{K}_\epsilon$ so that $\exists\ L(\theta_{\epsilon_{m1}}, k_{\epsilon_{m1}} \in \mathcal{L}_1^0(\Theta_\epsilon, \mathcal{K}_\epsilon) : \mid \theta - \theta_{m1} \mid < \epsilon/2$ and $\mid \rho_{m1} - \rho \mid < \epsilon/2$.

If $L(\theta_{\epsilon_{m1}}, k_{\epsilon_{m1}}) \in \mathcal{L}_1^0(\Theta_\epsilon, \mathcal{K}_\epsilon)$, then $L(\theta_{\epsilon_{m1}}, k_{\epsilon_{m1}}) \in \mathcal{L}_1^0(\Theta, \mathcal{K})\ \forall\ \Theta > \Theta_\epsilon$ and $\forall\ \mathcal{K} > \mathcal{K}_\epsilon$ by Note 3 above. Hence *1.* and *2.* are satisfied.♠

Let $\{\Theta_i,\ i = 1, 2, \ldots\}$ and $\{\mathcal{K}_i,\ i = 1, 2, \ldots\}$ represent the possible values of Θ and $\mathcal{K}$. Let $L(\theta^*_{m1}, k^*_{m1})$ be the best line in $\mathcal{B}_1$ and let $L(\theta^*_{i1}, k^*_{i1})$ be the best line in $\mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$.. We would like $\theta^*_{i1} \to \theta^*_{m1}$ and $k^*_{i1} \to k^*_{m1}$ (hence $\rho^*_{i1} \to \rho^*_{m1}$) as $i \to \infty$. To show this, we need one final result.

Define $\mathcal{C}_1 = \bigcup_{i=1}^{\infty} \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$. Note $\mathcal{B}_1 \subseteq \mathcal{C}_1$.

Theorem 4.1 For each $i = 1, 2, \ldots,$ let $L(\theta_{n_i}, k_{n_i}) \in \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$: $\theta_{n_i} \to \theta_{lim}$ and $k_{n_i} \to k_{lim}$ for some $\theta_{lim},\ 0 \le \theta_{lim} < \pi,$ and $k_{lim},\ 0 \le k_{lim} < \infty$. Let

$$\mathcal{D}_1 = \{L(\theta_{lim}, k_{lim}) : \exists \text{ a sequence } \{L(\theta_{n_i}, k_{n_i})\}_{i=1}^{\infty},\ L(\theta_{n_i}, k_{n_i}) \in \mathcal{L}_1^0(\Theta_i, \mathcal{K}_i) \text{ such that } \theta_{n_i} \to \theta_{lim} \text{ and } k_{n_i} \to k_{lim}\}$$

Then the best line in $\mathcal{B}_1$ is the best line in $\mathcal{D}_1$.

Proof: Note that $\mathcal{C}_1 \subseteq \mathcal{D}_1$ and $\mathcal{B}_1 \subseteq \mathcal{D}_1$. Since the minimal least squares line must pass through $(\bar{x}, \bar{y})$ and $(\bar{x}, \bar{y}) \in rect$, the best line in $\mathcal{B}_1 \equiv$ the best line in $\mathcal{D}_1$.♠

For the following claim, we assume the optimal line (i.e., the line which maximizes the fitness function or, in our case, minimizes the least squares error) is unique and that the$x$ in D. fitness function $f : [0, \pi] \times [-M, M]$ is continuous where, for any line in $\mathcal{D}_1$, the distance of the line from the origin is less than M.

Proposition 4.3 Let $L(\theta_{m1}^*, k_{m1}^*)$ be the best line in $\mathcal{D}_1$ and for each i, $i = 1, 2, \ldots$, let $L(\theta_{i1}^*, k_{i1}^*)$ be the best line in $\mathcal{L}_1^0(\Theta_i, \mathcal{K}_i)$. Then $\theta_{i1}^* \to \theta_{m1}^*$ and $k_{i1}^* \to k_{m1}^*$ as $i \to \infty$.

Proof: Let $L(\theta_{m1}^*, k_{m1}^*)$ be the best line in $\mathcal{D}_1$, i.e., if $L(\theta_{m1}, k_{m1}) \in \mathcal{D}_1$ and $L(\theta_{m1}, k_{m1}) \neq L(\theta_{m1}^*, k_{m1}^*)$ then $f(\theta_{m1}, k_{m1}) < f(\theta_{m1}^*, k_{m1}^*)$.

Let $E_i = \{(\theta_{m1}, k_{m1}) : \mathrm{d}((\theta_{m1}, k_{m1}), (\theta_{m1}^*, k_{m1}^*)) < 1/j_i\}$ where j_i is chosen so that if $(\theta_{i1}, k_{i1}) \in E_i$ and $(\theta_{j1}, k_{j1}) \in E_i^c$ then $f(\theta_{i1}, k_{i1}) > f(\theta_{j1}, k_{j1})$, and $j_i \to \infty$ as $i \to \infty$. Such sets E_i exist since the optimum is unique*.

Note that for each i, $\exists\ \epsilon_i > 0 :\ L(\theta_{i1}^*, k_{i1}^*) \in \mathcal{L}_1^0(\Theta_{\epsilon_i}, \mathcal{K}_{\epsilon_i}),\ |\ \theta_{i1}^* - \theta_{m1}^*\ | < \epsilon_i/2,$ $|\ \rho_{i1}^* - \rho_{m1}^*\ | < \epsilon_i/2$, and $(\theta_{i1}^*, k_{i1}^*) \in E_i$. The best line in $\mathcal{L}_1^0(\Theta_{\epsilon_i}, \mathcal{K}_{\epsilon_i}) \in E_i$, and the best line in $\mathcal{L}_1^0(\Theta, \mathcal{K})$, $\forall\ \Theta > \Theta_{\epsilon_i}$, $\forall\ \mathcal{K} > \mathcal{K}_{\epsilon_i}$, will also be in E_i (the sets E_i, $i = 1, 2, \ldots$, are nested sets).

Let the best line in $\mathcal{L}_1^0(\Theta_{\epsilon_i}, \mathcal{K}_{\epsilon_i})$ be $L(\theta_{\epsilon_i}^*, k_{\epsilon_i}^*)$. Note that $\theta_{\epsilon_i}^* \to \theta_{m1}^*$ and $k_{\epsilon_i}^* \to k_{m1}^*$ as $i \to \infty$. That is, if $L(\theta^*, k^*)$ is the best line in $\mathcal{L}_1^0(\Theta, \mathcal{K})$, then $\theta^* \to \theta_{m1}^*$ and $k^* \to k_{m1}^*$ as $\Theta \to \infty$ and $\mathcal{K} \to \infty$.♠

In the above proof it was stated that the sets E_i, $i = 1, 2, \ldots$, exist because the optimum is assumed to be unique. We now prove this.

Proposition 4.4 Define $E_i = \{(\theta_{m1}, k_{m1}) : \mathrm{d}((\theta_{m1}, k_{m1}), (\theta_{m1}^*, k_{m1}^*)) < 1/j_i\}$ where j_i is chosen so that if $(\theta_{i1}, k_{i1}) \in E_i$ and $(\theta_{h1}, k_{h1}) \in E_i^c$ then $f(\theta_{i1}, k_{i1}) > f(\theta_{h1}, k_{h1})$, and $j_i \to \infty$ as $i \to \infty$. Assume that $f : [0, \pi] \times [-M, M]$ is continuous and has a unique maximum. Then such sets E_i exist.

Proof: We prove this by contradiction. For each i, E_i constitutes an open disk containing $(\theta_{m1}^*, k_{m1}^*)$. From topology[17] we know that if $\mathcal{A}$ is any open set containing $(\theta_{m1}^*, k_{m1}^*)$ then $f(\mathcal{A}) \subseteq [f(\theta_{m1}^*, k_{m1}^*) - \epsilon, f(\theta_{m1}^*, k_{m1}^*) + \epsilon]$ for some $\epsilon > 0$.

So suppose not.

Then for all open sets $\mathcal{A}$ containing $(\theta^*_{m1}, k^*_{m1})$, if $(\theta_a, k_a) \in \mathcal{A}$ then $f(\theta_a, k_a) \leq f(\theta^c_a, k^c_a)$ for some point $(\theta^c_a, k^c_a) \notin \mathcal{A}$. But $f(E_i) \rightarrow f(\theta^*_{m1}, k^*_{m1})$ as $i \rightarrow \infty$, where

$$f(\theta^*_{m1}, k^*_{m1}) = \max_m \{f(\theta_m, k_m); L(\theta_m, k_m) \in \mathcal{D}_1, \ m = 1, 2, \ldots\}$$

and $(\theta^*_{m1}, k^*_{m1})$ is unique. Contradiction. ♠

4.2.1 Remarks

1. If we choose both Θ_{i_0} and $\mathcal{K}_{i_0}$ to be large, so that $\mid \theta_i - \theta_{i-1} \mid$ and $\mid k_i - k_{i-1} \mid$ are both small, then the maximal line $L(\theta^*_{i_0}, k^*_{i_0})$ will be close to the optimal line $L(\theta^*_{m1}, k^*_{m1})$.

2. In developing our genetic algorithm, it seems logical to start with an initial choice for $(\Theta_{i_0}, \mathcal{K}_{i_0})$ and run the algorithm for a finite number of iterations, resulting in an approximation $L(\Theta^{*a}_{i_0}, \mathcal{K}^{*a}_{i_0})$ of $L(\Theta^*_{i_0}, \mathcal{K}^*_{i_0})$. If Θ_{i_0} and/or $\mathcal{K}_{i_0}$ are/is small, it is possible for $L(\Theta^{*a}_{i_0}, \mathcal{K}^{*a}_{i_0})$ to be close to $L(\Theta^*_{i_0}, \mathcal{K}^*_{i_0})$ in terms of probability but not close to $L(\Theta^*_{i_0}, \mathcal{K}^*_{i_0})$ or $L(\theta^*_{m1}, k^*_{m1})$ in terms of Euclidean distance. Since it is unknown whether given values for Θ and $\mathcal{K}$ are 'small' or 'large', we will start by searching, given $(\Theta_{i_0}, \mathcal{K}_{i_0})$, for a $L(\Theta^{*a}_{i_0}, \mathcal{K}^{*a}_{i_0})$ that is close to $L(\Theta^*_{i_0}, \mathcal{K}^*_{i_0})$ in terms of probability, and then choose subsequent $(\Theta_i, \mathcal{K}_i)$ so that our approximations $L(\Theta^{*a}_i, \mathcal{K}^{*a}_i)$ move closer to $L(\theta^*_{m1}, k^*_{m1})$ in terms of Euclidean distance.

We now would like to extend this theory to the case where the number of lines $k_0 > 1$, k_0 known.

4.3 Case $k_0 = n_0$, n_0 known, $n_0 > 1$

We consider using genetic algorithms to fit the minimum least-squares k_0-piecewise linear function to the data set $(\boldsymbol{x}, \boldsymbol{y})$ where $k_0 = n_0$, n_0 known. The fitted values $\hat{y}_{0m}$ of the optimal function satisfy

$$(\textstyle\sum_{i=1}^{n_0} \sum_{m=N_{0(i-1)}}^{N_{0i}-1} (\hat{y}_{0m} - y_m)^2)^{-1} = \max_j \{(\sum_{i=1}^{n_0} \sum_{m=N_{j(i-1)}}^{N_{ji}-1} (\hat{\hat{y}}_{jm} - y_m)^2)^{-1} :$$

$$\hat{\mathbf{y}}_j = L(\theta_{jn_0}, \mathbf{k}_{jn_0}), \ L(\theta_{jn_0}, \mathbf{k}_{jn_0}) \in \mathcal{L}^0_{n_0}\} \qquad (10)$$

where

$$L(\theta_{jn_0}, \mathbf{k}_{jn_0}) \mid_x = L(\theta_{ji n_0}, k_{ji n_0}) \mid_x \text{ for } x_{N_{j(2(i-1))}} \leq x \leq x_{N_{j(2i-1)}}, \ x_{N_{j(0)}} = x_{(1)},$$
$$x_{N_{j(2n_0-1)}} = x_{(N)}, \ x_{N_{j(i+1)}} = x_{N_{j(i)}} + 1 \text{ for } i = 1, \ldots, 2(n_0 - 1).$$

Note that $\mathbf{x}_{N_j} = \{x_{N_{j(0)}}, x_{N_{j(1)}}, \ldots, x_{N_{j(2n_0-1)}}\}$ depends on the function $L(\theta_{jn_0}, \mathbf{k}_{jn_0})$.

Our search space is

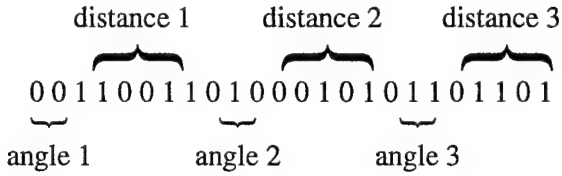
$\mathcal{L}^0_{n_0} = \{L_{jn_0}(\boldsymbol{x}) : L_{jn_0}(\boldsymbol{x}) \in \mathcal{L}_{n_0},$
$L_{j_i n_0}$ is of the form $x\cos\theta_{ji} + y\sin\theta_{ji} = l_{\theta_{ji}} + k_{ji}\delta \ \ \forall \ \ i = 1,\ldots,n_0,$
$\theta_{ji} \in \{0, \frac{\pi}{2^{l_a}}, \frac{2\pi}{2^{l_a}}, \ldots, \frac{(2^{l_a}-1)\pi}{2^{l_a}}\}$, $k_{ji} \in \{0,1,\ldots,2^{l_d}-1\}$, and $\delta = diag/(2^{l_d}-1)\}$.

We will only consider those $L_{jn_0}(\boldsymbol{x}) \in \mathcal{L}_{n_0}$:

1. $-\cos\theta_{j(i)}/\sin\theta_{j(i)} \neq -\cos\theta_{j(i+1)}/\sin\theta_{j(i+1)}$ $\forall\, i = 1,\ldots,n_0$ (no adjacent parallel lines).
2. Let $z_{(j1)},\ldots,z_{(j(n_0-1))}$ be the intersection points of $L_{jn_0}(\mathbf{x})$. Then $x_{N_{j(2i-1)}} \leq z_{(ji)} \leq x_{N_{j(2i)}}$ for $i = 1,\ldots,(n_0-1)$.

As before, let $L_{jn_0}(\boldsymbol{x})$ be denoted as $L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})$.

The theory for the case $k_0 = 1$ can be extended to this case. Each string can be designed to represent an n_0-piecewise function $L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0}) \in \mathcal{L}_n$ satisfying the above assumptions; note that each string will resemble a combination of n_0 strings from the $k_0 = 1$ case. For example, if $n_0 = 3$, $l_a = 3$, and $l_d = 5$, then a string may look like



We then employ a similar GA optimization procedure to find the string which represents the minimal least-squares n_0-piecewise function.

The optimization procedure can alternatively be viewed as a two-step process: for each possible choice of $\mathbf{x}_{N_j}$, say, $\mathbf{x}^i_{N_j}$, find the optimal choice for $L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})$, say, $L^{\dagger_i}(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})$. Then, from the set of all functions $\{L^{\dagger_i}(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})\}_{i\geq 1}$, select the optimal function, say, $L^{\ddagger}(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})$. If we let $\{\mathbf{x}^i_{N_j}\}_{i\geq 1}$ be the set of possible values for $\mathbf{x}_{N_j}$ and let $\{L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0}))\}_i$ denote the set of all n_0-piecewise functions whose pieces intersect in such a way that $\mathbf{x}^i_{N_j}$ satisfies the above, then

$$f(L^{\ddagger}(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})) = \max_i\{\max\{f(L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0}));\ L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0}) \in \{L(\boldsymbol{\theta}_{jn_0}, \boldsymbol{k}_{jn_0})\}_i\}\} \tag{11}$$

4.4 Further Remarks and Discussion

4.4.1 Fitness Function

In equation (10) the fitness function for our genetic algorithm was stated as

$$(\sum_{i=1}^{n_0} \sum_{m=N_{0(i-1)}}^{N_{0i}-1} (\hat{y}_{0m} - y_m)^2)^{-1}$$

If all of the data points fall on a line (or on several linear segments), however, it is possible for $\sum_{i=1}^{n_0} \sum_{m=N_{0(i-1)}}^{N_{0i}-1} (\hat{y}_{0m} - y_m)^2$ to equal zero. To avoid this case the above fitness function may be modified by the addition of an arbitrary positive constant ϵ, yielding

$$((\sum_{i=1}^{n_0} \sum_{m=N_{0(i-1)}}^{N_{0i}-1} (\hat{y}_{0m} - y_m)^2) + \epsilon)^{-1}$$

4.4.2 Assumptions

In the above theory, we have assumed that the optimal number of lines is known. However, in most cases, the optimal number of lines is unknown and must be estimated. It may be possible to generalize the above theory to this case by utilizing a genetic algorithm which allows for variable string lengths. Then, given a data set, the algorithm could select the optimal number of pieces (from an initial set of possible values) as well as the optimal piecewise function.

4.4.3 Curve Fitting

In this paper we have only considered the fitting of piecewise linear functions. It is well known that piecewise linear functions can be used to approximate a curve to any degree of accuracy. Hence curve fitting can be seen as a generalization of the above problem. Suppose our interest was in fitting the optimal curve to a data set. An approach to this problem may be to apply the above theory, given a set of points, to find the piecewise linear function which best approximates the optimal curve. The quality of the approximation would be influenced by the number of pieces as well as the number of iterations.

4.4.4 More than 2 Dimensions

In two dimensions, our interest is in fitting a k_0-piecewise linear function to a data set $\{(x_1, y_1), \ldots, (x_N, y_N)\}$. A similar problem exists for data in d_0 dimensions, $d_0 > 2$; namely, fitting a k_0-piecewise hyperplane to a data set $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{id_0})$ for $i = 1, \ldots, N$. To solve this problem using genetic algorithms, we could consider extending the methods outlined above as follows:

Each string would represent an individual solution to the problem, i.e., a k_0-piecewise hyperplane in d_0 dimensions. From geometry we know that a hyperplane in $\mathcal{R}^{d_0}$ may be represented as

$$x_{iN}\cos\theta_{N-1} + \gamma_{N-1}\sin\theta_{N-1} = c$$

where

- $(x_{i1},\ldots,x_{iN})$ is a point on the hyperplane
- $\gamma_{N-k} = x_{i(N-k)}\cos\theta_{N-(k+1)} + \gamma_{N-(k+1)}\sin\theta_{N-(k+1)}$ for $k = 1,\ldots,N-1$
- θ_{N-k} is the angle that the projection of the normal to the $(X_1 - \cdots - X_{N-(k-1)})$ plane makes with the $X_{N-(k-1)}$ axis for $k = 2,\ldots,N-1$
- θ_{N-1} is the angle that the projection of the normal to the hyperplane makes with the X_N axis
- θ_0 is the angle that the projection of the normal to the X_1 plane makes with the X_1 axis ($\theta_0 = 0$), and
- c is the perpendicular distance of the hyperplane from the origin.

To specify c we use the hyper-rectangle *hrect* containing the points $\{(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_N,\mathbf{y}_N)\}$, just as we used the rectangle containing the points $(\mathbf{x},\mathbf{y})$ to specify d in $\mathcal{R}^2$. Let $\mathrm{l_a}$ be the number of bits used to specify an angle, and $\mathrm{l_d}$ be the number of bits used to specify c. Let H_{ji} represent the ith hyperplane, or piece, of the jth k_0-piecewise hyperplane. Then for a given set of angles $\Theta_{ji} = (\theta_{ji_0},\ldots,\theta_{ji_{N-1}})$, $\theta_{ji_m} \in \{0,\ldots,\frac{(2^{l_a}-1)\pi}{2^{l_a}}\}\ \forall\ m = 0,\ldots,d_0-1$, we let $c = l_{\Theta_{ji}} + k_{ji}\delta$ where $l_{\Theta_{ji}}$ is the minimum distance of the origin from one of the hyperplanes passing through a vertex of *hrect*, *diag* be the maximum diagonal of *hrect*, $k_{ji} \in \{0,1,\ldots,2^{l_d}-1\}$, and $\delta = diag/(2^{l_d}-1)$. Let $\Gamma_{ji} = (\gamma_{ji_1},\ldots,\gamma_{ji_{N-1}})$. Then our discrete search space $\mathcal{H}^0_{k_0}$, like $\mathcal{L}^0_{k_0}$, may be written as

$$\begin{aligned}\mathcal{H}^0_{k_0} = \{H_{jk_0}(\mathbf{x}_1,\ldots,\mathbf{x}_N) : H_{jk_0}(\mathbf{x}_1,\ldots,\mathbf{x}_N) \in\ & \mathcal{H}_{k_0},\ H_{ji}k_0 \text{ is of the form}\\ & x_{iN}\cos\theta_{ji_{(N-1)}} + \gamma_{ji_{(N-1)}}\sin\theta_{ji_{(N-1)}} = l_{\Theta_{ji}} + k_{ji}\delta\ \ \forall\ \ i = 1,\ldots,k_0,\\ & \mathbf{\Gamma}_{ji} \text{ and } \Theta_{ji} \text{ are as specified above, } \theta_{ji_m} \in \{0,\ldots,\tfrac{(2^{l_a}-1)\pi}{2^{l_a}}\}\\ & \forall\ m = 0,\ldots,d_0-1,\ k_{ji} \in \{0,1,\ldots,2^{l_d}-1\},\ \text{and } \delta = diag/(2^{l_d}-1)\}\end{aligned}$$

We now use GA's to search $\mathcal{H}^0_{k_0}$ for the k_0-piecewise hyperplane which minimizes the least-squares distance of the points $(\mathbf{x}_i, y_i)$ from the hyperplane.

5 Implementation and Results

5.1 Case $k_0 = 1$

5.1.1 Data

The data set used is from Weisberg's text, *Applied Linear Regression*[18]. The set contains 17 data points that were collected in an experiment by James D. Forbes, a Scottish physicist, designed to study the relationship between atmospheric pressure (in Hg.) and boiling point (F°).

5.1.2 Genetic Algorithm

We used a fixed population size of $M = 10$ and a string length of $L = 20$, with 8 bits representing θ and 12 bits representing k; note that once l_{theta} and δ are known, specifying k is equivalent to specifying d. The single-point crossover probability, p, was fixed at 0.8. The mutation probability q varied with the iteration number over a range of [0.0015, 0.5], either increasing or decreasing depending on the value of $Nit/Nmax$, where Nit is the current iteration number and $Nmax$ is the maximum number of iterations. $Nmax$ was set at 1500, at which time the maximum fitness value attained and it's corresponding string were reported. As stated previously, for a given string S_j and the fitted values $\hat{y}_{ji}$ of the line it represents, the fitness value is given as

$$f(\hat{\mathbf{y}}_\mathbf{j}) = (\sum_{i=1}^{N}(\hat{y}_{ji} - y_i)^2)^{-1} \tag{12}$$

The results of the GA were compared to the results of a simple linear regression program designed to fit the least squares line to the data.

5.1.3 Experimental Results

The proposed algorithm was tested on the data described in Section 5.1.1. The results are shown in Table 1 and Figure 4. The results of the GA are comparable to the results of the least-squares regression program. The disparity between the results of the two methods may be the result of, for example, the algorithm failing to converge (due to an insufficient value for $Nmax$) or lack of precision in the results of the GA (due to insufficient string length).

	Nit	function	$\max_j f(\hat{\mathbf{y}}_\mathbf{j})$
Approx	1500	$\hat{f}(x) = 0.882x - 39.32$	0.450
Actual		$f(x) = 0.895x - 42.14$	0.464

Table 1: Results of Experiment 1

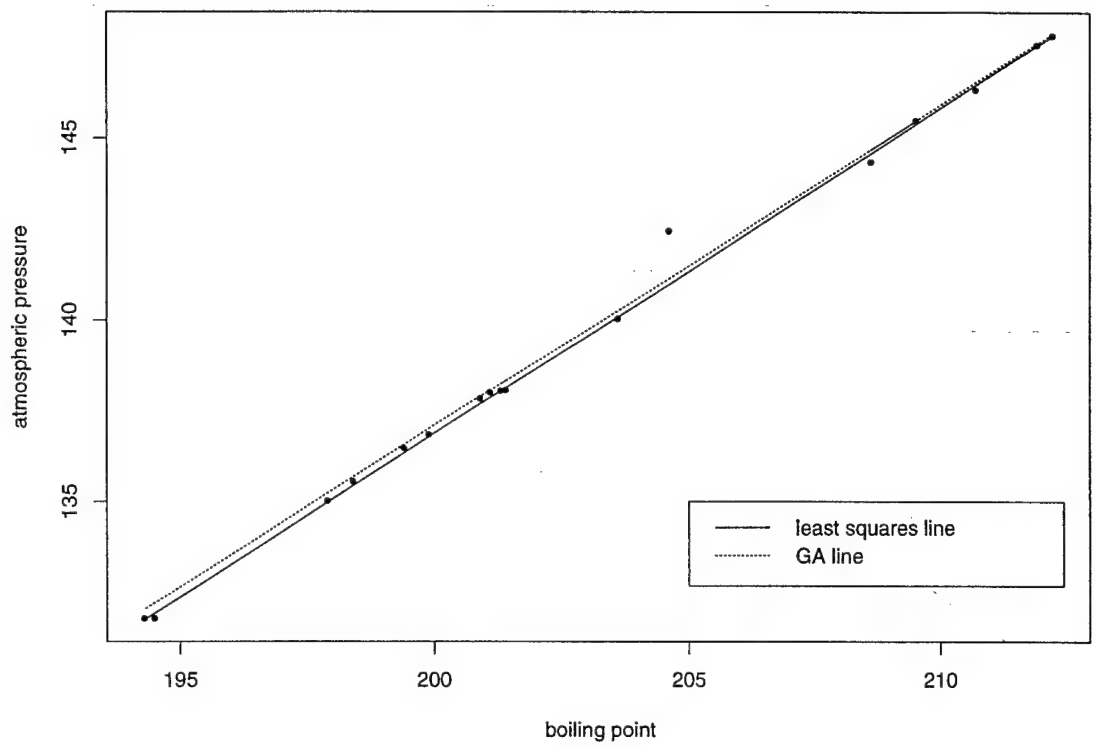


Figure 4: Results of Experiment 1

5.2 Case $k_0 = n_0$, n_0 known, $n_0 > 1$

We will demonstrate this case when $k_0 = 3$.

5.2.1 Data

An artificial data set was created by first selecting a 3-piecewise linear generating function $g(x)$ whose value is given by

$$g(x) = \begin{cases} 2+x & -1 \leq x < 0 \\ 2-x & 0 \leq x < 1 \\ x & 1 \leq x \leq 2 \end{cases} \tag{13}$$

where $\mathbf{x} = (-1, -0.96, -0.92, \ldots, 1.96, 2)$. For each value $x_i \in \mathbf{x}$, the corresponding vector $\mathbf{y}_i = (y_{i1}, \ldots, y_{i5})$ consists of 5 values randomly generated from a Normal $(g(x_i), 0.1)$ distribution.

5.2.2 Genetic Algorithm

Each string or chromosome represents a 3-piecewise linear function

$$L(\theta_{j3}, \mathbf{k}_{j3}) = (L(\theta_{j_1 3}, k_{j_1 3}), L(\theta_{j_2 3}, k_{j_2 3}), L(\theta_{j_3 3}, k_{j_3 3}))$$

over the range of $\mathbf{x}$. Let $z_{(j1)}$ be the intersection point of $L_{j_1 3}$ and $L_{j_2 3}$ and let $z_{(j2)}$ be the intersection point of $L_{j_2 3}$ and $L_{j_3 3}$. We chose to consider only those piecewise functions L_{j3} for which

- $z_{(j1)}$ and $z_{(j2)}$ exist (no adjacent parallel pieces)
- $x_{(1)} \leq z_{(j1)} \leq z_{(j2)} \leq x_{(N)}$

The fitness value for a given string is then

$$(\sum_{i=1}^{3} \sum_{m=N_{j(2(i-1))}}^{N_{j(2i-1)}} (\hat{y}_{jm} - y_m)^2)^{-1} \tag{14}$$

where $(N_{j0}, \ldots, N_{j5})$:

- $x_{N_{j0}} = x_{(1)}$, $x_{N_{j5}} = x_{(N)}$
- $x_{N_{j1}} \leq z_{(j1)} \leq x_{N_{j1}+1} = x_{N_{j2}}$; $x_{N_{j3}} \leq z_{(j2)} \leq x_{N_{j3}+1} = x_{N_{j4}}$

and

$$\hat{y}_{jm} = \begin{cases} L(\theta_{j_1 3}, k_{j_1 3})|_x & x_{N_{j0}} \leq x \leq x_{N_{j1}} \\ L(\theta_{j_2 3}, k_{j_2 3})|_x & x_{N_{j2}} \leq x \leq x_{N_{j3}} \\ L(\theta_{j_3 3}, k_{j_3 3})|_x & x_{N_{j4}} \leq x \leq x_{N_{j5}} \end{cases} \tag{15}$$

5.2.3 Design Modifications

With $n_0 = 3$, it became evident that if we set $\mathrm{l_a} = 8$ and $\mathrm{l_d} = 12$ as above, so that each string had length $L = 60$, the size of the population matrix and the number of iterations required for convergence would make our approach computationally expensive. To avoid this problem, the genetic algorithm was divided into *hierarchical loops*. The modified algorithm can be described as follows:

- Set the global parameters $M = 40$, $p = 0.8$, and Nit = number of iterations per loop = 3000.
- Loop 1
    1. Choose $\mathrm{l_a} = 2$ and $\mathrm{l_d} = 5$ ($L = 21$) so that the 4 angles $a_1, \ldots, a_4$ and 10 k values $k_1, \ldots, k_{10}$ that can be represented are evenly spaced over the ranges $[\pi/4, \pi]$ and $[0, 2^5 - 1]$, respectively.

2. Generate the initial population $\mathcal{Q}$ so that all strings represent functions which meet the above specifications for $z_{(j1)}$ and $z_{(j2)}$.
3. Execute a genetic algorithm beginning with $\mathcal{Q}$ to find the optimal string $S_1 = (s_{1a_1}, s_{1d_1}, \ldots, s_{1a_3}, s_{1d_3})$.
4. Create matrices S_{opA} and S_{opD} with three rows, where row i represents the optimal angle or optimal distance of piece i, $i = 1, \ldots, 3$. Place the appropriate sections of S_1 into S_{opA} and S_{opD}.

- Loop 2

5. Generate a new matrix Q^*, also with $\mathrm{l_a} = \mathrm{f2}$ and $\mathrm{l_d} = 5$, so given that a_i and k_i are the optimal angle and distance most recently selected for piece i, the 4 possible angles and 10 possible k values for piece i represented in Q^* are evenly spaced over the ranges $[a_i - \pi/(4^2), a_i + 2\pi/(4^2)]$ and $[(2(k_i) - 1)/2, k_i + 1]$.
6. Repeat step 3 to find $S_2 = (s_{2a_1}, s_{2d_1}, \ldots, s_{2a_3}, s_{1d_3})$.
7. Place the appropriate sections of S_2 into S_{opA} and S_{opD} (now $S_{opA_i} = (s_{1a_i}, s_{2a_i})$ and $S_{opD_i} = (s_{1d_i}, s_{2d_i})$).

- For loop j, $\mathrm{j} \geq 3$ repeat steps 5-7 where angle and distance values are now evenly spaced over $[a_i - \pi/(4^j), a_i + 2\pi/(4^j)]$ and $[(2(k_i) - 1)/2, k_i + 1]$.
- When the desired degree of precision has been reached, the algorithm is stopped and the matrices S_{opA} and S_{opD} contain the optimal piecewise function.

Note that the size of the population matrix remains constant regardless of the number of loops being performed. Hence the use of this modified version of a GA avoids the manipulation of large matrices, reducing the required computational resources, without adversely affecting the precision of the resulting solution.

5.2.4 Experimental Results

Table 2 shows the performance of the proposed GA based algorithm on the artificial data described in Section 5.2.1. The fitness value for the generating lines is stated for purpose of comparison.

	Nloops	function	$\max_j f(\hat{\mathbf{y}}_\mathrm{j})$
Approx	2	$\hat{f}(x) = \begin{cases} x + 1.996 & -1 \leq x < 0 \\ 1.991 - x & 0 \leq x < 1 \\ x - 0.019 & 1 \leq x \leq 2 \end{cases}$	0.266836
Generator		$g(x) = \begin{cases} x + 2 & -1 \leq x < 0 \\ 2 - x & 0 \leq x < 1 \\ x & 1 \leq x \leq 2 \end{cases}$	0.2633

Table 2: Results of Experiment 2

After 2 loops and only 6000 iterations, the GA converged to a 3-piecewise function with a greater fitness value than the original generating lines.

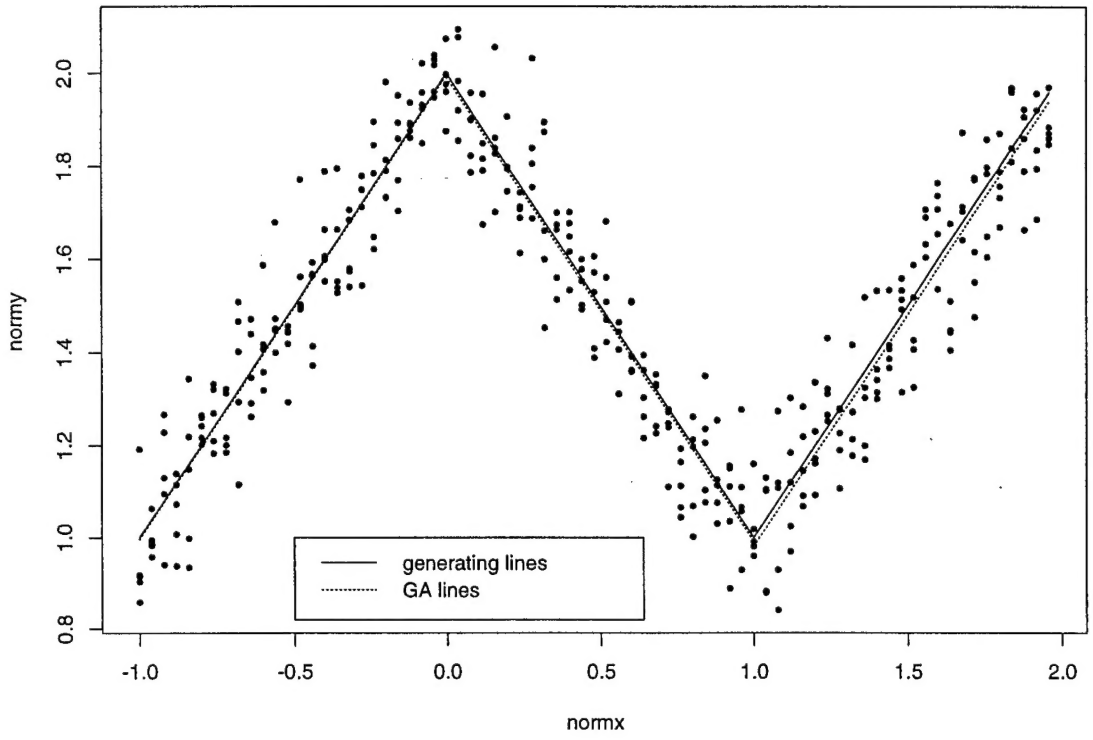


Figure 5: Results of Experiment 2

6 Conclusions and Future Research

We have conducted 2 experiments employing GA's for the fitting of piecewise linear functions to datasets in $\mathcal{R}^2$. Our results demonstrate that GA's can yield near optimal results at limited computational expense.

These encouraging results have suggested several directions for future research. Our experiments involve cases where the number of lines is known. We would like to design an algorithm which determines the optimal number of lines as well as their placement. Many interesting problems involve data sets of more than 2 dimensions; hence we would like to explore the use of GA's for fitting hyperplanes and other multidimensional surfaces. The comparison of a multivariate GA for surface fitting with existing methods, such as MARS[20,21] and projection pursuit regression[22,23] is certainly worth investigating.

The research of Ms. J. Pittman is supported by the Army Research Office under Assert grant # DAAG55-97-1-0219.

The research of C.A. Murthy is supported by the army research office under Assert grant # DAAHO4-96-1-0082.

References

[1] Larry L. Schumaker, *Spline Functions: Basic Theory.* John Wiley & Sons, New York(1981).

[2] B. Cheng and D.M. Titterington, "Neural networks: A review from a statistical perspective", *Statistical Science*, **9**, 2(1994).

[3] Michael Werman and Z. Geyzel, "Fitting a second degree curve in the presence of error", *IEEE Trans. on Pattern Anal. and Machine Intelligence*, **17**, 207-211(1995).

[4] H.J. Larson, "Least squares estimation of linear splines with unknown knot locations", *Computational Statistics and Data Analysis*, **13**, 1-8(1992).

[5] H. Schwetlick and T. Schütze, "Least squares approximation by splines with free knots", *BIT*, **35**, 361-384(1995).

[6] Randall L. Eubank, *Spline Smoothing and Nonparametric Regression.* Marcel Dekker, New York(1988).

[7] Leo Breiman, comment to Bing Cheng and D.M. Titterington, "Neural networks: A review from a statistical perspective", *Statistical Science*, **9**, 2(1994).

[8] S.K. Pal, S. Bandyopadhyay, and C.A. Murthy,"Genetic algorithms for generation of class boundaries", *IEEE Trans. Syst. Man Sybern.*, accepted.

[9] C.A. Murthy, S. Bandyopadhyay, and S.K. Pal, "Genetic algorithm-based pattern classification: Relationship with bayes classifier". In *Genetic Algorithms for Pattern Recognition*, eds. S.K. Pal and P.P Wang, CRC Press, Boca Raton(1996).

[10] S. Bandyopadhyay, C.A. Murthy, and S.K. Pal, "Theoretical performance of genetic pattern classifier", *IEEE Trans. Syst. Man Sybern.*, communicated.

[11] S.K. Pal and P.P. Wang, Eds. *Genetic Algorithms for Pattern Recognition.*CRC Press, Boca Raton(1996).

[12] David S. Chen and Ramesh C. Jain, "A robust back propagation learning algorithm for function approximation", *IEEE Trans. on Neural Networks*, **5**, 467-479(1994).

[13] K. Warwick and R. Craddock, "An introduction to radial basis functions for system identification a comparison with other neural network methods", *Proc. of the IEEE Conf. on Decision and Ctrl*, **1**, 464-469(1996).

[14] F.M. Aparicio Acosta, "Radial basis functions and related models: An overview", *Signal Processing*, **45**, 37-58(1995).

[15] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press(1996).

[16] D. Bhandari, C.A. Murthy, and S.K. Pal, "Genetic algorithm with elitist model and its convergence", *Int. J. Patt. Recog. Art. Intell.*, accepted.

[17] J. Dugundji, *Topology.* Allyn and Bacon, Boston, MA(1966).

[18] S. Weisberg, *Applied Linear Regression, 2nd Edition.* John Wiley & Sons, New York(1985).

[19] Charles J. Stone, "The use of polynomial splines and their tensor products in multivariate function estimation", *Ann. Statist.*, **22**, 118(1994).

[20] Jerome H. Friedman, "Multivariate adaptive regression splines", *Ann. Statist.*, **19**, 1(1991).

[21] C. Hung, "Estimation of a projection-pursuit type regression model", *Ann. Statist.*, **19**, 142(1991).

[22] C. Posse, "Projection pursuit exploratory data analysis", *Computational Statistics and Data Analysis*, **20**, 669(1995).

[23] R.K. Beatson and G.N. Newsam, "Fast evaluation of radial basis functions: I", *Computers Math. Applic.*, **24**, 7-19(1992).

[24] H.N. Mhaskar and Charles A. Micchelli, "Approximation by superposition of sigmoidal and radial basis functions", *Advances in Applied Mathematics*, **13**, 350-373(1992).

[25] C.B. Roosen and T.J. Hastie, "Automatic smoothing spline projection pursuit", *J. of Comp. and Graph. Statist.*, **1**, 235-248(1992).

[26] M.H. Mhaskar, "Neural networks for optimal approximation of smooth and analytic functions", *Neural Computation*, **8**, 164-177(1996).

[27] J.A. Leonard, M.A. Kramer, and L.H. Ungar, "Using radial basis functions to approximate a function and its error bounds", *IEEE Trans. on Neural Networks*, **3**, 624-627(1992).

[28] Gábor Lugosi and Kenneth Zeger, "Nonparametric estimation using neural networks", *IEEE Inter. Symp. on Info. Theory - Proc.*, 112(1994).

[29] C.L. Karr, D.A. Stanley, and B.J. Scheiner, "Genetic algorithm applied to least squares curve fitting", Report of investigations # 9339, U.S. Dept. of the Interior, Bureau of Mines, 1991.

[30] T.M. Apostol, *Mathematical Analysis, 2nd Edition.* Addison-Wesley, Reading, Mass(1974).

[31] J. Stewart, *Calculus.* Wadsworth, Belmont, CA(1986).